Received 14 May 2024; revised 30 July 2024; accepted 7 August 2024. Date of publication 14 August 2024;
date of current version 26 August 2024. The review of this article was arranged by Associate Editor Xiang Sun.

Digital Object Identifier 10.1109/OJCS.2024.3442921

Publish Subscribe System Security Requirement: A Case Study for V2V Communication

HEMANT GUPTA AND AMIYA NAYAK (Senior Member, IEEE)

School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON K1N 6N5, Canada

CORRESPONDING AUTHOR: AMIYA NAYAK (e-mail: nayak@uottawa.ca).

This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

ABSTRACT The Internet of Things (IoT) enables the linkage between the physical and digital domains, with wireless sensor networks (WSNs) playing a vital role in this procedure. The market is saturated with an abundance of IoT devices, a substantial proportion of which are designed for consumer use and have restricted power and memory capabilities. Our analysis found that there is very little research done on defining the security requirements of the IoT ecosystem. A crucial first step in the design process of a secure product entails meticulously scrutinizing and recording the precise security requirements. This paper focuses on defining security requirements for Vehicle-to-Vehicle (V2V) communication systems. The requirements are specified utilizing the Message Queuing Telemetry Transport for Sensor Network (MQTT-SN) communication protocol architecture, specifically tailored for use in sensor networks. The modified Security Requirement Engineering Process (SREP) and Security Quality Requirement Engineering (SQUARE) methodologies have been used in this paper for the case study. The security of the communication between the ClientApp and the road-side infrastructure is our main priority.

INDEX TERMS Internet of Things (IoT), MQTT-SN, security requirements, Vehicle-to-Vehicle (V2V) communication.

I. INTRODUCTION

Currently, there is a wide range of IoT devices accessible in every residence. Hence, numerous multinational enterprises worldwide are actively creating IoT devices, services, and technologies to gain a competitive edge in the market beforehand. However, they fail to regard security as a functional necessity, resulting in a decrease in the priority of security issues [1]. Hence, organizations exhibit hesitancy in adequately implementing security measures for their equipment and services. When Transmission Control Protocol/Internet Protocol (TCP/IP) was initially developed, the significance of security was not widely recognized, and the majority of individuals were unaware of the potential exploits that attackers could carry out through security vulnerabilities. Due to these factors, TCP/IP was not well built for security purposes. Due to the insecure architecture, attackers can exploit numerous vulnerabilities, resulting in security breaches and significant financial losses. Nevertheless, we possess a comprehensive understanding of the importance of security and the precise capabilities that attackers might exploit through security vulnerabilities. Therefore, it is imperative to engage in discussions about IoT security throughout the development of IoT-related standards to prevent the recurrence of past errors as modifying systems becomes increasingly challenging as the device reaches an advanced level of development, resulting in greater effort and cost. Hence, it is imperative to prioritize security right from the inception of any product development.

Most developers and engineers prioritize attaining security objectives above delineating precise actions to be executed. For an IoT product, it is crucial to provide clear specifications (i.e., actions) together with functional requirements (desired outcome) to allow a security architect or designer to efficiently deal with any security issues. These particular characteristics are commonly known as security requirements. As far as we are aware, there is a dearth of research specifically focused on the security needs of the IoT environment. Many academics have mainly focused on threat models and targeted those threats. The process being used so far has a gap in understanding the importance of every critical and non-critical asset and the priority and risk involved in addressing the security requirements. As it has been said, no system is 100% secure. Therefore, we need a proper structural security requirement-gathering framework that helps us make an informed and risk-calculated decision.

Similarly, an indispensable necessity for V2V (Vehicle-to-Vehicle) communication is a specialized framework for gathering security requirements. This is because of the distinctive obstacles and safety-critical characteristics of this technology. Vehicle-to-vehicle (V2V) communication raises particular security considerations, including the need for identification, data integrity, confidentiality, and protection against threats like spoofing or jamming [2]. In addition, V2V systems depend on intricate wireless communication protocols such as DSRC or C-V2X, which require specific security measures tailored to these protocols [3], [4]. The necessity for a specific security architecture is emphasized by the real-time nature of V2V communication, which requires timely and reliable delivery of safety-critical information. Furthermore, V2V systems are required to adhere to regulatory standards and meet interoperability needs across a wide range of vehicle fleets. Ensuring privacy protection is essential in V2V communication due to the transmission of sensitive information regarding the individuals within the vehicles [5], [6], [7]. Implementing a specialized framework for gathering security requirements can methodically identify and handle specific issues, leading to strong security measures that protect road safety and user privacy in V2V communication systems.

Security requirements are defined as constraints on system functions. The requirements should explicitly specify the objects that require protection and provide a clear rationale for why such protection is necessary. Essentially, requirements should encompass the specifications regarding the individuals who have the authority to perform certain actions and the specific timeframes in which these actions can be carried out. Furthermore, they should serve the purpose of clearly defining what actions are permissible or forbidden for each individual involved. The security requirements articulate the security objectives in operational terms that are sufficiently detailed to be communicated to the system's designer/architect (actions). During the early 2000s, numerous researchers were developing various methodologies or approaches to establish the security prerequisites for software systems [8]. Among numerous methodologies, two techniques that exhibit significant similarities have been identified: SREP [9] and SQUARE [10].

SREP and SQUARE are better choices for security requirement gathering due to their tailored focus on security, systematic approach, consideration of contextual factors, integration with existing processes, community support, and comprehensive coverage of security concerns [9], [10], [11]. By employing these frameworks, organizations can effectively mitigate security risks and enhance the resilience of IoT deployments against evolving threats.

1) These frameworks focus explicitly on identifying and prioritizing security concerns, which can be paramount in IoT deployments.
2) These frameworks are guidelines, methodologies, and tools that assist in identifying, analyzing, and documenting security requirements across Software Development Life Cycle (SDLC) stages. Following a structured process, organizations can achieve complete coverage of all potential threats, reducing the chances of missing crucial vulnerabilities that IoT systems often conceal.
3) IoT deployments are characterized by extremely varied environments, usage scenarios, and stakeholders, all with distinct and relevant security requirements and limitations. Both SREP and SQUARE are explicit regarding the importance of context.
4) SREP and SQUARE frameworks are expertly designed to seamlessly integrate with existing software engineering processes, making them versatile and adaptable to different development methodologies and organizational contexts. Whether organizations follow traditional waterfall models or agile approaches, these frameworks can be confidently incorporated into the requirements engineering phase. This ensures that security considerations are addressed early in the development life cycle, providing a sound foundation for secure software development.
5) SREP and SQUARE offer comprehensive coverage of security concerns, encompassing various security dimensions such as confidentiality, integrity, availability, authentication, authorization, and non-repudiation.

A. CONTRIBUTIONS

V2V communication is an important aspect of today's autonomous industry. In this paper, our contributions are:

- Our research focuses on defining security requirements for one of the main aspects of V2V communication, i.e., sending secure distress signals in an emergency, which has not been addressed with a proper framework.
- The detailed modified SREP and SQUARE method [12] to design the threat model for vehicle communication using publish/subscribe protocol defined in [13], MQTT-SN (i.e., MQTT for Sensor Network) [14] have been defined. In [12], our main focus was on security requirement for smart lock. Here, more details of the modified method with reasoning and steps involved in defining security requirements for the V2V communication use case have been provided.
- Our focus has been on the weakest link of the MQTT-SN communication protocol, i.e., the link between the client and the gateway device via the MQTT-SN forwarder, which many researchers did not address for the MQTT-SN protocol.

B. OUTLINE

The subsequent sections of the paper are arranged in the following manner. Section II offers background and related work on security requirements, publish-subscribe system MQTT-SN and vehicle communication security. Section III offers an overview of modified SREP (Security Requirements and Evaluation Process) and SQUARE (Security and Quality Requirements Engineering) specifically tailored for IoT devices. Section IV explains the detailed procedure for gathering security requirements for V2V communication using the MQTT-SN publish/subscribe protocol. Sections V and VI present the analysis of the collected security needs and the resulting conclusion, respectively.

II. BACKGROUND & RELATED WORKS

A. SECURITY REQUIREMENTS

A security requirement is a statement that specifies a condition or constraint that must be satisfied by a system or software application to ensure its security. These requirements are crucial for identifying, mitigating, and managing security risks throughout the development life cycle of a system or application. Security requirements encompass various aspects of security, including confidentiality, integrity, availability, authentication, authorization, and non-repudiation. The background of two security requirement-gathering methods are discussed below:

- The SREP is a crucial process that ensures security considerations are integrated into the system design from the early stages, minimizing the risk of vulnerabilities and ensuring the confidentiality, integrity, and availability of the system's assets. The SREP process is a structured and systematic process that aims to identify, analyze, and specify security requirements for a software application or system. The process involves a series of steps, including initiation, elicitation, analysis, specification, validation, verification, and maintenance. Collaboration and communication among stakeholders, including security experts, software developers, system architects, and end-users, are crucial throughout the SREP process. The SREP process is iterative and should be integrated into the overall software development lifecycle to achieve a secure and resilient system. By following the SREP process, organizations can ensure that their systems and applications are secure and can meet the ever-changing security threats, regulations, or organizational needs.
- The SQUARE is a structured methodology for incorporating security considerations into the software development process. It aims to ensure that the final product meets the desired security goals while reducing the risk of security breaches and vulnerabilities. The SQUARE method is iterative and can be integrated into existing software development processes, such as the waterfall or agile methodologies. It involves seven phases, including problem definition, elicitation, classification, analysis, specification, validation, verification, and documentation. SQUARE emphasizes the need for collaboration and communication among stakeholders, including security experts, developers, architects, and end-users, to ensure that security is adequately addressed from the early stages of software development. By following the SQUARE method, organizations can enhance the security of their software systems and reduce the risk of security breaches and vulnerabilities.

The related work with respect to the security requirements are discussed here:

Michael et al. [15] focused on business profile modelling to express security requirements like trust, confidentiality and integrity on an abstract level. They use a model-driven approach to define security goals and their related requirements. However, there is no proper description of threats, their actors and the risks involved for the designers to make a proper informed decision.

Zhao et al. [16] focus on the overall architecture of Solar Insecticidal Lamsps Internet of Things (SIL-IoT) and share the potential security scenarios and attacks and prospective solutions. However, they are missing a properly designed threat model to identify the security requirements more clearly and attackers and vulnerable points in all channels of communication.

Myagmar et al. [17] demonstrate the significance of threat modelling in establishing security requirements. Furthermore, they emphasize that no system can be completely safeguarded. Consequently, the designer must carefully consider all potential risks and determine which ones to prioritize addressing, while also accepting the risk associated with some aspects of the system. Nevertheless, all the conducted case studies only focus on software tools that do not cater to IoT devices.

Firesmith [18] asserts that the majority of designers and scholars tend to focus on defining architecture and design limitations, rather than addressing genuine security issues. The author delineated many categories of security prerequisites and disseminated the guidelines and protocols to enable requirements engineers to accurately specify them without impeding the security and functionality of the system. The author further emphasizes that there are security criteria that can be reused [19] and established a framework for precisely outlining the security requirements. Nevertheless, the above actions are insufficient in addressing the risks associated with the IoT ecosystem.

Mellado et al. [9] explain the SREP is a methodical and organized procedure that seeks to identify, assess, and define security requirements for a system or software application. The Security Requirement Engineering (SRE) process incorporates security considerations into the system design from its inception, thereby reducing the likelihood of vulnerabilities and guaranteeing the confidentiality, integrity, and availability of the system's assets. The following is a background summary of the typical steps involved in the Security Requirement Engineering process: initiation, elicitation, analysis, specification, validation, verification, and maintenance. The SRE process is iterative and should be integrated into the overall software development life cycle to achieve a secure and resilient system. The SREP design here is not focused on the IoT ecosystem.

Mead et al. [10] define the SQUARE process as a methodology that specifically targets the identification and definition of security quality requirements for software systems. The software development process is enhanced by a systematic method that integrates security considerations. The objective is to guarantee that the end product fulfills the intended security objectives. Here is a background summary of the SQUARE method: problem definition, elicitation, classification, analysis, specification, validation, verification and documentation. The SQUARE method is iterative and can be integrated into existing software development processes, such as the waterfall or agile methodologies.

The system explained by Firesmith, SQUARE and SREP, do not cover the steps and guidelines to address the security requirements of the IoT ecosystem and the risk involved with each requirement.

Huang et al. [20] construct a resilient security framework and examine the security prerequisites by scrutinizing several scenarios such as body IoT, home IoT, and hotel IoT. In addition, a compelling survey is carried out by a group of 30 adults to ascertain the hierarchy of various security criteria, namely confidentiality, integrity, availability, and access control.

Rahman et al. [21] present a comprehensive IoT security framework consisting of four layers and analyze various security concerns. Furthermore, the authors examine the security prerequisites for each component (namely, IoT sensor node, base station, network, cloud) in their cloud scenario.

Lee et al. [22] discuss the security risks and requirements associated with the IoT open platform and networked ecosystem. They emphasize the significance of ensuring security protocols are compatible with existing network infrastructure and can operate effectively in diverse network environments.

Abomhara et al. [23] examine the vision, security threats, and problems of IoT. The major security criteria chosen are authentication, secrecy, and data integrity. Alqassem et al. [24] classify security requirements into access control, data integrity, contextual integrity, intrusion detection, and non-repudiation.

Kim et al. [25] categorize the various types of IoT devices and examine the security risks associated with each type. Kim's research focuses on several essential security requirements, such as the creation of efficient protocols and encryption algorithms, ensuring secure communication, safeguarding data, enhancing physical security, establishing device identification and permission systems, and implementing device monitoring and control mechanisms.

Oh et al. [26] defined the basic security requirements by using three IoT characteristics and six elements of IoT. They provide a comparison with other researchers working on IoT security requirements. However, they have not considered the threat model and designed a proper framework for defining security requirements.

B. PUBLISH-SUBSCRIBE SYSTEM

Many distributed IoT systems require more flexible communication systems. Synchronous and point-to-point systems are inflexible and motivate a more flexible and loosely coupled communication system; therefore, publish-subscribe systems are becoming more popular.

Publish-subscribe is a communication architecture where senders of the messages are not aware of the receivers of the messages, and messages are categorized into classes. Publish-Subscribe system [27] is based on the subject/topic or content (i.e., keyword) matching or location-based, which we have explained in later sections. The primary advantage of the publish-subscribe system is the support for asynchronous processing. The message-sending node (i.e., publisher) does not need to know about the nodes receiving the messages. Similarly, the message-receiving node (i.e., subscriber) does not need to know about the nodes sending the messages. The subscribers register the subject/topic or pattern for which they want to get the information to the broker. The publishers publish information on a subject/topic or pattern to the broker. If the topic or pattern matches, the message is forwarded to the subscriber by the broker.

The publish-subscribe systems are classified primarily using two schemes, i.e., the subscription and architectural models [28]. In the subscription model, a subscriber can define which published messages they want to receive (i.e., subscription). The broker checks if the published message matches the subscription, and then the message is forwarded by the broker to the subscriber. There are two types: topic-based and content-based subscription models.

The communication architecture is an essential part of the publish-subscribe system as it is also related to the subscription model. The architecture defines how the nodes are organized in the system and communicate with each other. There are two main architectures as defined in [28]: broker and peer-to-peer architecture.

The publish-subscribe system related work will be discussed below:

Liu et al. [28] surveyed eight different publish-subscribe systems and provided a taxonomy for comparing them based on category (i.e., content-based, subject-based, hybrid), system architecture (i.e., client-server, peer-to-peer), matching algorithm, multicasting algorithm and language support. In addition, the authors tried to define the general security goals for content-based publish-subscribe systems, but it was not used in their taxonomy. Gryphon [29] is the only mentioned publish-subscribe system that provides client authentication and access control mechanisms.

Pelz [27] introduced a new method of measuring the quality of the publish-subscribe system, i.e., quality of robustness (QoR) (i.e., number of nodes allowed to fail without data loss). The author tested the system with zero node failure to all nodes failure. The author found that management node (i.e., broker) failure was the worst case for QoR. Therefore, the author suggested that the IoT system designer always ensure that the management node never fails so that there is less damage to clients.

Wang et al. [30] tried to define the general security requirements and issues present in the content-based publish-subscribe system. The authors listed the security requirements as confidentiality, integrity and availability and sub-divided them. The authors also mentioned which security requirements after implementation would take away the benefits of the publish-subscribe system. Finally, the authors provided an overview of security architecture for the large-scale publish-subscribe system.

Lagutin et al. [31] introduced a security design for a publish-subscribe network. They used elliptic curve cryptography and certificates to address the integrity, availability and scalability in forwarding (i.e., publisher/subscriber) and rendezvous (i.e., broker) planes. In addition, they implemented filters based on security roles such as name space owner, scope owner, publisher, and data source in the pub/sub architecture to avoid bogus subscription requests. In their design, publishers could not publish under the topic until at least one subscription request was present for that topic.

Ammar et al. [32] surveyed the security aspects of eight different IoT frameworks that would help develop industrial and consumer-based IoT applications. Only four of the IoT platforms support pub/sub messaging protocols (i.e., AWS IoT by Amazon, Kura by Eclipse, ARM Mbed and Azure IoT by Microsoft), and the security of these platforms depends on symmetric cryptography algorithms or Transport Layer Security (TLS). The authors also mentioned threats, which are mitigated by the available security features.

As far as we know, no literature thoroughly discusses the process of determining security requirements and the associated risks for IoT devices and services that use a publish-subscribe architecture. The primary objective of solution providers is to identify and mitigate potential threats. However, it is widely acknowledged that achieving absolute security is an unattainable goal. Hence, it is imperative to accurately define security requirements and assess the corresponding risks to make well-informed decisions regarding which security requirements necessitate immediate attention. The main objective of this paper is to establish security requirements for two specific scenarios using the modified SREP and SQUARE methodology.

C. MQTT FOR SENSOR NETWORK (MQTT-SN)

MQTT-SN [14] was designed in 2013 specifically for wireless sensor networks (WSN), which cannot communicate over TCP. It communicates over User Datagram Protocol (UDP), Zigbee and Bluetooth Low Energy (BLE). MQTT-SN message consists of two parts: a 2- or 4-octet long header and an optional variable part. There are three components of MQTT-SN:

1) *MQTT-SN Clients:* These are any battery-operated constrained devices [33]. These can be a publisher or a subscriber who wants to send/receive a message to/from the broker.
2) *MQTT-SN Gateway:* It is a bridge between MQTT-SN clients and the MQTT broker. It might be integrated with the MQTT broker as well. There are two types of MQTT-SN Gateways:
   a) Transparent Gateway: It will set up and maintain an MQTT connection between each MQTT-SN client and the broker. There will be the same number of connections between the gateway and broker as MQTT-SN clients connected to the gateway.
   b) Aggregating: It will have only one connection between the gateway and the broker. All messages coming from different MQTT-SN clients stop at the gateway, and then the gateway decides which information will be forwarded to the broker.
3) *MQTT-SN Forwarder:* The MQTT-SN client can also access the gateway through the forwarder in case the gateway is not directly attached to the WSN. The forwarder only encapsulates the MQTT-SN client message it receives from the wireless side and forwards it to the gateway without modification; similarly, it decapsulates the message received from the gateway and forwards it to the MQTT-SN clients. In the process of encapsulation, the forwarder adds broadcast radius and wireless node id (i.e., a node which has sent or should receive the encapsulated MQTT-SN message).

The important parameters in the CONNECT, PUBLISH and SUBSCRIBE control packets are:

1) *ClientID:* It is a unique identifier used by clients. The broker uses it to identify the client and their state relating to an ongoing MQTT-SN session.
2) *Clean Session:* It is a flag in the CONNECT message. When set to false, it means the client wants to have a persistent session (keeps track of all the session's information as the devices go offline or disconnect), which will help the client if it connects with the same ClientID.
3) *Topics:* In MQTT-SN, "pre-defined" topic IDs (i.e., Topic alias) and short topic names are introduced, for which no registration is needed. The topic IDs in MQTT-SN are the strings that the broker uses to filter messages for each connected client. Topics can be predefined by the manufacturers and brokers and may also be defined by the publishers. MQTT-SN Topics have one or more levels, separated by a forward slash (/). A client can subscribe to an exact topic used by the publisher, or a client can use the wildcard. There are two types of wildcards:
   - *Single level wildcard (+):* Replaces one topic level. For example, both */Myhouse/kitchen/temperature and /Myhouse/room1/temperature* can be represented as */Myhouse/+/temperature*.
   - *Multi-level wildcard (#):* Replaces multiple topic levels. With the help of multi-level wildcard, a client receives all messages of a topic that begins with the pattern before the wildcard character. Using the

above example, and it can be written as */Myhouse/# or /#*.

The MQTT-SN and its security related work are discussed below:

Kao et al. [34] proposed a connection authentication technique that is both simple and secure. The technique is based on MQTT-SN, AEAD (ChaCha20-Poly1305), hash function, digital signature (ECDSA), key exchange (ECDHE), and hash function. These components are used to create Safe MQTT-SN, a secure version of MQTT-SN that enables clients to encrypt and decrypt published messages. Kao et al. replaced RSA with the ChaCha20-Poly 1305 algorithm, which has the ability to authenticate published messages and has been proven to have lower power consumption than AES-GCM.

Roldán-Gómez et al. [35], [36] analyzed the security aspects of the MQTT-SN protocol due to its susceptibility to various vulnerabilities that can be exploited. To exploit vulnerabilities, they suggested several attacks with basic scenarios using Contiki and Cooja. Roldán-Gómez et al. conducted measurements to assess the impact of the attacks and put forward a threat detection system utilizing the Complex Event Processing (CEP) engine. This system achieved an impressive F1 score of 0.9963 and is capable of identifying attacks in an IoT context.

Asmaa [37] approached employing transmission flags, which can authenticate all aspects of data exchange across networks and gather and examine data from numerous devices. Although the payload length is short, this method has demonstrated its effectiveness in assessing the degree of security of data transfer, which necessitates encryption for tracking purposes.

Farahani [38] have shown an effort to intercept wireless communication by employing ZigBee and implementing AES-CSM. The authors have made an attempt to use the same key twice in order to achieve this, such as by deliberately causing a node reset. To execute a denial-of-service attack, they suggested modifying the content of ZigBee packets, regardless of encryption, to introduce a flaw that disrupts the intended functioning of the protocol.

Husnain et al. [39] identified 81 vulnerabilities associated with MQTT using National Vulnerability Database (NVD) and Common Vulnerabilities and Exposures (CVE), which are widely recognized mechanisms for reporting vulnerabilities. The vulnerabilities arise from three primary issues: inadequate packet length, absent essential fields, and a deficiency in logical error checks, along with other miscellaneous faults that do not fit into the aforementioned categories. The authors outline a comprehensive approach for detecting and preventing vulnerabilities in an end device. This approach consists of five steps: protocol intuition, vulnerabilities assessment, protocol identification, protocol parsing, and stringent protocol validations. Additionally, rules are defined to enhance the effectiveness of the vulnerability detection and prevention process. Their technology effectively detects and prevents these vulnerabilities from being exploited on IoT devices.

Sochor et al. [40] explained the reflection attack in their study, which exploits the MQTT-SN protocol and leads to DRDoS attacks, where many servers are targeted. Only a minuscule fraction of 0.02% of the victim's bandwidth is required to initiate a reflection assault that fully saturates the target system's bandwidth. To address this issue, the authors suggested a technique that entailed incorporating an additional handshake into the protocol and restricting Quality of Service (QoS) for levels 0 and 1. A four-way handshake (consisting of the PUBLISH, PUBREC, PUBREL, and PUBCOMP messages) can be utilized to ensure QoS level 2, which guarantees that each message is received only once. However, because this was not enough, implementing a three-way handshake would have been a better choice for the client to communicate with the broker.

Sadio et al. [41] proposed a security system for the MQTT-SN protocol that is lightweight and relies on the chacha20-poly1305 AEAD. This system guarantees encryption of nodes from end to end. The system utilizes various pre-existing Arduino libraries for the MQTT-SN client. As the number of nodes increases, the only additional requirement is secret key management. Chen et al. [42] explored the root causes of MQTT security issues and proposes various offensive and defense strategies, including machine learning-based security, replay attacks, man-in-the-middle attacks, anomaly detection, and more. With the increasing complexity of IoT, there is a greater need for effective means to deal with attacks and anomalies. The author hopes to establish a smart, safe, and reliable IoT infrastructure using technology based on data analysis. Practical attack tests in various scenarios are needed for future MQTT security research.

Andy et al. [43] analyzed the MQTT protocol utilized in IoT devices, revealing its shortcomings in terms of data privacy, authentication, and data integrity. The paper delves into the reasons behind the insufficient security measures in many IoT systems, illustrating various attack scenarios that can compromise the protocol. For instance, one scenario entails an attacker scanning a public network and carrying out denial of service attacks or data manipulation. Another scenario involves packet data sniffing and modification within the local network. The paper suggests the implementation of security mechanisms like TLS or ECC to ensure data confidentiality and non-repudiation. Nonetheless, security measures for resource-constrained devices still require further development.

Several academics are now studying the security of MQTT and MQTT-SN, but they are not considering the gateway and forwarder. They continue to regard end-device communication on the Internet as a distinct matter.

D. V2V COMMUNICATION SECURITY

V2V communication involves the transmission of information between vehicles in order to improve road safety, efficiency, and convenience. V2V communication facilitates the exchange of data between cars, including details like speed, position, acceleration, and direction. This enables surrounding vehicles to proactively anticipate and respond to potential dangers immediately. Although V2V communication shows potential to enhance road safety and traffic control, it is crucial to prioritize the security and privacy of these conversations to prevent malicious attacks and unauthorized access. It is primarily focused on safeguarding the integrity, confidentiality, and authenticity of the data communication. This involves guaranteeing that communications transmitted between vehicles are impervious to interception, tampering, or unauthorized imitation, therefore upholding trust and dependability in the V2V communication system.

V2V communication commonly uses wireless communication protocols like Dedicated Short Range Communications (DSRC) or Cellular Vehicle-to-Everything (C-V2X) and satellite communication. These protocols establish the criteria and details for exchanging data between automobiles and infrastructure components. These protocols incorporate security features to protect V2V communications from different security risks. However, as far as we know, very little research is available related to the security of vehicles sending distress signals. Alongside the available research, there is no proper infrastructure or guidelines followed in determining the complete security requirements and methodology to address and determine the threats to V2V communication. Therefore, this gap shows the need for our research.

Here, we discuss here some related work done in the field of V2V communication security:

Muslam conducted a comprehensive literature review on V2V communication [26]. The main objectives of the review were to identify existing security protocols, assess their strengths and weaknesses, investigate the integration of emerging technologies such as 5G and edge computing, analyze the interaction between V2V communication protocols and broader IoT frameworks, and evaluate the impact of existing or proposed standards on the implementation of secure V2V communication systems.

Jabeen et al. [44] conducted a comprehensive examination of the security and privacy implications of connected vehicles, as well as an assessment of cloud platforms. Through a comparative examination of multiple cloud platforms, we can choose the most effective cloudlet for obtaining improved performance.

Cao et al. [45] proposed an efficient communication framework for on-the-move electric vehicle (EV) charging application, based on the P/S mechanism and public buses to disseminate the condition information of charging stations (CSs). The CS selection decision on where to charge is made by individual EVs for privacy and scalability benefits. The authors promote the concept of Mobility as a Service (MaaS) to exploit the mobility of public transportation vehicles such as buses to bridge the information flow to EVs, given their opportunistic encounters. The authors demonstrate the advantage of introducing buses as mobile intermediaries for information dissemination based on a common EV charging management system under the Helsinki city scenario. The authors further study the feasibility and benefit of enabling EVs to send their charging reservations for CS selection logic via opportunistically encountered buses. Results show that this advanced management system improves both performances at the CS and EV sides. Another research from Cao et al. [46] proposed another efficient communication framework for on-the-move EV charging applications based on the publish/subscribe mechanism. They use push and pull mode to send and gather data from the road-side units. The evaluation results showed that the decreased number of times EVs obtain information from RSUs deteriorates the performance in terms of average waiting time as well as information freshness from the EV perspective.

Taghizadeh et al. [47] introduced a novel V2V operation that empowers electric vehicle owners with excess battery energy to transfer it to other vehicles during their daily commutes. This innovative system facilitates energy transfer between two EV on-board chargers, allowing EV owners to exchange stored energy without impacting the grid. Additionally, an IoT platform that employs the MQTT protocol is developed to manage interactions between EV owners and chargers. By utilizing the untapped energy in EV batteries, this V2V operation can alleviate grid strain.

Papadimitratos [2] discussed the security and privacy mechanisms needed for vehicular communication systems. It emphasizes the importance of scalable credential management and convergence on policies that determine pseudonym lifetimes, rates of change, CRL update, and distribution requirements. The text also highlights the need for cryptographic primitives beyond those in standards and literature, such as the Elliptic Curve Digital Signature Algorithm (ECDSA), and the introduction of post-quantum cryptographic primitives. It also mentions the importance of countering clogging denial of service attacks and cooperative validation, especially as network data rates and security levels increase. Finally, the text emphasizes dedicated solutions for vehicle maneuver coordination to detect insider attacks and effectively detect subtle and sophisticated attacks in real-time without misclassifying actual, benign maneuvering as misbehavior.

Based on our knowledge of the security of V2V communication, we have discovered that the existing security architecture is still riddled with numerous areas of weakness and vulnerabilities. The security architectures proposed in the above literature for MQTT, MQTT-SN, and vehicle communication directly mentioned the threats. However, there is no proper framework to identify the security requirements and the risks involved with each threat. One person cannot resolve all the vulnerabilities, which shows the need for a framework to identify and prioritize these security requirements to calculate the risk involved in case we are unable to address any particular security requirement.

III. PROPOSED SECURITY REQUIREMENT METHODOLOGY

In this section, the modified SREP and SQUARE framework steps needed for the IoT domain and the proper threat-defining framework to properly analyze a threat are defined.

A. MODIFIED SREP AND SQUARE

SREP and SQUARE are inadequate in tackling the IoT domain because they do not adequately address its distinct difficulties, which include varied ecosystems, scale, contextual considerations, resource restrictions, dynamic nature, and transdisciplinary requirements. Their disadvantages include a lack of scalability, flexibility, and advice specifically designed for the complexity of IoT, such as the presence of diverse devices, limited resources, and the need for real-time interactions. Specialized frameworks are necessary to adapt to the dynamic surroundings, regulatory issues, and interdisciplinary nature of IoT, in order to effectively gather security requirements. Therefore, modified SREP and SQUARE methodologies can be utilized for the advancement and implementation of contemporary Smart IoT devices. Given the escalating intricacy and security apprehensions linked to IoT devices, it is imperative to integrate solid security requirements engineering methods. The utilization of SREP can be applied to ascertain and delineate the security goals, hazards, and prerequisites that are distinct to IoT devices. SQUARE, however, is especially valuable for specifying security-related quality criteria in IoT devices. To optimize the SREP and SQUARE for modern Smart IoT devices, the following improvements should be taken into account [12]:

1) Examine the specific hazards and vulnerabilities associated with IoT: Enumerate the distinct hazards and perils linked to IoT devices, including but not limited to unauthorized entry, data breaches, device manipulation, and privacy apprehensions. Integrate these hazards into the risk assessment and analysis phases of SREP.
2) Discuss communication protocols and interfaces: IoT devices commonly utilize many protocols and interfaces for communication. Make sure that SREP and SQUARE take into consideration the security of these communication routes, encompassing encryption, authentication, and safe data transfer.
3) Incorporate factors related to the IoT ecosystem: IoT devices are commonly integrated into a broader ecosystem comprising cloud services, gateways, and networked devices. Broaden the extent of SREP and SQUARE to encompass the security needs of the entire ecosystem, encompassing the interactions between devices and cloud services.
4) Privacy concerns arise due to the fact that IoT devices frequently handle and communicate confidential personal information. Integrate privacy standards and considerations into the process of engineering security requirements, taking into account adherence to data protection rules, implementation of user permission processes, and utilization of data anonymization techniques.
5) Enhance the emphasis on authentication procedures and access control for the IoT devices. Integrate prerequisites for ensuring the secure verification of devices, the authorization of users, and the implementation of access control regulations that are customized to the unique requirements of IoT devices.
6) Physical security considerations: IoT devices are susceptible to physical attacks or tampering due to their physical accessibility. Specify the necessary measures for physical security, such as the inclusion of tamper detection technologies, the implementation of a secure enclosure design, and the provision of protection against physical theft or manipulation.
7) Revise and enhance the techniques used for evaluating the security of systems: Modify security testing approaches and processes to specifically cater to the distinct attributes of IoT devices. Examine vulnerability scanning, penetration testing, and fuzz testing that are specifically tailored to IoT protocols and interfaces.
8) Consistent surveillance and regular updates: IoT devices may necessitate ongoing surveillance to detect security flaws and updates to counteract evolving threats. Include provisions for continuous security monitoring, patch management, and firmware/software updates within the security requirements.

By adapting SREP and SQUARE to incorporate these IoT-specific factors, we can guarantee that the security requirements engineering process effectively tackles the problems and hazards linked to Smart IoT devices.

Their steps to account for the unique challenges and complexities of IoT environments have been modified. Here's a proposed framework:

1) *Identify Stakeholders and Assets:*
   - Identify all stakeholders involved in the IoT ecosystem, including manufacturers, developers, end-users, and regulatory bodies.
   - Identify the assets within the IoT system, such as devices, sensors, actuators, data repositories, and communication channels.
2) *Define Security Quality, Goals and Objectives:*
   - Define specific security goals and objectives tailored to IoT deployments, considering factors like data privacy, device integrity, resilience to cyberattacks, and compliance with industry regulations.
   - Identify security quality attributes relevant to IoT deployments, such as confidentiality, integrity, availability, authentication, authorization, and non-repudiation.
   - Consider additional quality attributes specific to IoT, such as resilience, privacy, scalability, and energy efficiency.
3) *Define Security Requirements Baseline:*
   - Define a baseline of security requirements based on industry best practices, standards, guidelines, and regulatory requirements applicable to IoT deployments.
   - Consider existing security frameworks and methodologies tailored to IoT, such as the IoT Security Foundation's Best Practice Guidelines.
4) *Analyze Threats and Vulnerabilities:*

- Conduct a thorough analysis of potential threats and vulnerabilities specific to IoT environments, including device tampering, data breaches, unauthorized access, and denial-of-service attacks.
- Consider both technical and non-technical threats, such as physical security risks and supply chain vulnerabilities.
- Use IoT-specific use cases and threat models to guide the elicitation process and ensure comprehensive coverage of security concerns.

5) *Elicit Security Requirements:*
   - Use various elicitation techniques, such as interviews, surveys, workshops, and brainstorming sessions, to gather security requirements from stakeholders.
   - Focus on requirements related to authentication, encryption, access control, secure firmware updates, secure bootstrapping, and secure communication protocols tailored to IoT deployments.
6) *Prioritize Security Requirements:*
   - Prioritize security requirements based on their criticality, impact on system functionality, and feasibility of implementation in IoT environments.
   - Consider factors like resource constraints, interoperability, scalability, and regulatory compliance when prioritizing requirements.
7) *Document Security Requirements:*
   - Document security requirements in a clear, concise, and structured manner, using standardized formats and templates.
   - Ensure that security requirements are traceable, measurable, and verifiable, enabling effective implementation, testing, and validation.

B. MODIFIED THREAT DEFINING FRAMEWORK FOR IOT

The threat-defining framework has been designed after defining modified security requirement-gathering methods. As numerical values for threat probability, severity and attacker's difficulty lack credibility, most practical risk assessment are based on qualitative ratings. Therefore, we use a categorical rating (low, medium, high) to calculate the risk [48]. In designing the threat model, the following framework have been used:

1) Threat: A threat is a potential cause of an unwanted incident, which may harm a system or organization, often accompanied by a condition avoiding the threat.
2) Attack Name: A successful threat is known as an attack.
3) Attacker's Difficulty/ Threat Difficulty: It defines the knowledge and tools an attacker needs to attempt a successful attack.
   a) Low: An attacker does not need any high-level technical knowledge of hacking tools or systems. The basic knowledge of how the system works is sufficient.
   b) Medium: An attacker needs to have basic knowledge of reverse engineering IoT devices and the structures of the protocol used in the communication.
   c) High: An attacker needs to have complete knowledge of hacking and has tools to intercept messages, modify them and send them back to the broker.
4) Threat Severity: The qualitative analysis of the threat is done based on asset and success on impact:
   a) Low: A threat severity is low when the successful impact does allow an attacker to make the device inaccessible.
   b) Medium: A threat severity is medium when the successful impact does allow an attacker to intercept messages and determine their contents.
   c) High: A threat severity is high when the successful impact allows an attacker to intercept messages, determine their contents, and respond with modified data, and the recipient accepts it as legitimate.
5) Threat Probability/Threat Likelihood: Based on level of attack difficulty, the probability of an attack occurs successfully and how easily it can be repeated are defined:
   a) Low: With the high level of attacker difficulty, it is less probable for the success of an attack and not able to repeat it often.
   b) Medium: With a medium level of attacker difficulty, it is probable for the success of an attack and as we are only capturing data or passive eavesdropping.
   c) High: With a low level of attacker difficulty, it is highly probable for the success of an attack, and an attacker may repeat it often.
6) Summary: It summarizes the whole threat and its objective.
7) Precondition: It defines the knowledge that an attacker should have for a successful attack.
8) Step: It defines different steps taken by a user, an adversary, and a broker to make a threat into a successful attack.
9) Postcondition: It defines the results after a successful attack.

C. GENERALIZED SECURITY REQUIREMENTS DEFINITION

In this section, the security requirements definitions for publish-subscribe model used for V2V communication will be formulated. All definitions of the terms we are going to use here in this document have been covered and derived from [48] and [49]:

1) Identification: The statement defines the degree to which an application must recognize its external components prior to engaging with them. The system should adhere to the privacy rules, which may necessitate the anonymity of users. It can also be explained as: who you say you are (i.e., name, user identifier etc.), what

you have taken (i.e., hardware key, smart card, identification card, etc.) and who you are (i.e., bio prints or behavioural traits).

2) Authentication: It ensures that person or identity are actually who or what they claim to be. It depends on identification. In order to handle the authentication requirements, we should have answers of at least one of these questions: who you know (i.e., mother maiden name, name of pet, etc.), what you have (i.e., taken, hardware key, smart card, identification card etc.), and who you are (i.e., bio prints or behavioural traits).
   The following three sub-divisions of authentication have been created:
   a) Data Origin Authentication: It should instill confidence in the authenticity of the data source. A subscriber should possess the capability to authenticate the publisher (i.e., sender) of the message.
   b) Device Authentication: It should ensure that the equipment involved in communication is truly what it claims to be, thereby instilling confidence in its identification. Within the pub-sub model, a broker possesses the exceptional capability to accurately identify and authenticate a device, hence preventing any other device from falsifying the authentication credentials of another device.
   c) Device Owner/User Authentication: It should instill assurance that the user/owner of the device is indeed who they claim to be. Within the pub-sub framework, it is imperative for the broker to possess the capability to authenticate the device owner in order to mitigate any complications arising from the transfer of device ownership.
3) Confidentiality: It is a property where non-authorized entities are not allowed to access plaintext messages.
   a) Information Confidentiality: It signifies that data is safeguarded in the event that the broker's reliability is questionable. Feasible, yet necessitates an agreement outside the usual communication channels between the publisher and subscriber. The out-of-band agreement negates the major advantages of the pub/sub architecture [48]. The pub/sub system is designed to resemble point-to-point models rather than many-to-many ones. There are two possible types. The initial data transfer between the publisher or subscriber and the broker is encrypted using point-to-point encryption. However, the data is accessible in plain text form at the publisher, subscriber, and broker. The second data is encrypted throughout its entire journey until it reaches the intended recipients, ensuring its confidentiality. The data remains securely encrypted at the broker. To achieve complete information confidentiality, it is necessary to establish a group key distribution between the publisher and subscribers through an out-of-band method. The dissemination of keys beyond the main communication channel eliminates the advantages of the fundamental publish/subscribe model and transitions to a multicast model.
   b) Message Header Confidentiality: It means sending message header securely from one point to another.
4) Integrity: Any creation, modification or deletion done by unauthorized parties in the data and communication will be detected.
   a) Payload Integrity: The content of the message should remain unchanged after it has been sent by the sender. In the pub-sub system, the message transmitted by the publisher must remain unchanged until it is received by the subscribers.
   b) Header Integrity: The Header must be safeguarded against unauthorized alterations. The assumption is that the publisher/subscribers have confidence in the broker. For example, when a user subscribes to topic A in the pub-sub system, they will get a message that has been published under topic A. It is imperative that the records of topics that a certain subscriber is interested in remain unmodifiable by anyone.
   c) Broker Integrity: Compromising the trusted broker jeopardizes the integrity of the entire system. The term "trusted broker" refers to a situation where the broker has access to the data sent by the publisher to the subscribers in plaintext, ensuring point-to-point confidentiality.
5) Availability: It is the property according to which resources and services remain accessible for authorized use. It limits the frequency and size of publication. Customized publication control allows subscribers to specify which publisher can publish information. Subscribers can use a secret, challenge-response system. It will help get rid of the bogus notifications.
6) Authorization: It is the property of computing resources being accessible by authorized entities. Authorization is achieved through access control. A broker, publisher or subscriber may control which subscribers or publishers may access the information.
7) Physical Protection: It specifies the extent to which an application or center shall protect itself from physical assault. Publisher and subscriber should protect their physical device or application from tampering.

IV. SECURITY REQUIREMENTS ON V2V COMMUNICATION BASED ON MQTT-SN

The main security objectives based on the need for the V2V communication use case are identified below.

1) Authentication: Verify the legitimacy of the end user and the source of the data.
2) Confidentiality: Ensure the prevention of unauthorized disclosure of information.

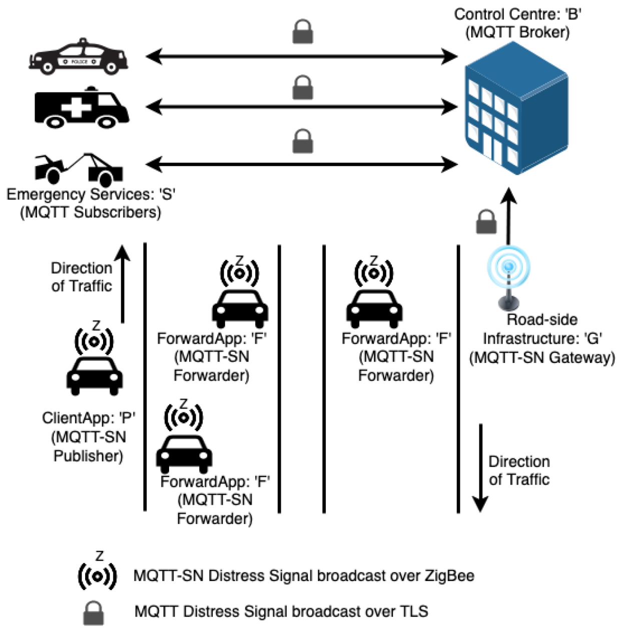
Control Centre: 'B' (MQTT Broker)
Emergency Services: 'S' (MQTT Subscribers)
Direction of Traffic
ClientApp: 'P' (MQTT-SN Publisher)
ForwardApp: 'F' (MQTT-SN Forwarder)
ForwardApp: 'F' (MQTT-SN Forwarder)
ForwardApp: 'F' (MQTT-SN Forwarder)
Road-side Infrastructure: 'G' (MQTT-SN Gateway)
Direction of Traffic
MQTT-SN Distress Signal broadcast over ZigBee
MQTT Distress Signal broadcast over TLS

FIGURE 1. **Vehicle-to-Vehicle Communication using MQTT-SN over Zigbee [13].**

3) Integrity: Ensure that information cannot be modified without authorization.
4) Accessibility: Ensure that both the information and the device are easily accessible.
5) Authorization: Guarantee that information and devices can only be accessed by individuals who have been granted permission.

A. V2V USE CASE SCENARIO

V2V communication is a rising research area. It is used for many reasons, such as informing about road congestion, traffic blockage, and accidents and sending distress signals to ask for help in case of potential and immediate danger exists. We have seen that many methods have been applied; however, the only concern with those solutions is that they need cellular or satellite networks to communicate. What happens when no such network is available or end-users cannot afford such upgrades in their vehicles? To tackle this problem, the MQTT-SN communication model has been proposed here, which uses the publish-subscribe method to send a distress signal from a vehicle to emergency services. A distress signal is sent using MQTT-SN over ZigBee [13]. Fig. 1 shows our design architecture for distress signals moving from a vehicle to emergency services. The design architecture enables distress signals to move from a vehicle to emergency services using a ClientApp or vehicle on-board unit. The distress signal data contains relative location information and vehicle information, which is sent using VIN (Vehicle Identification Number) as ClientID. The distress message is broadcasted to the Control Centre using the MQTT-SN Forwarder and roadside infrastructure (i.e., MQTT-SN gateway). Based on topic and subscription, the signal is forwarded to the respective emergency services by the Control Centre. The message contains the VIN and the user's last known location information, which helps emergency services to dispatch the help.

In this document, when we mention the security of V2V communication and different attacks on the use case, we mainly refer to the attackers exploiting communication protocol to achieve the desired goal. When it comes to emergency services, there may and may not be human risk factors involved, and therefore, it is important to make sure these signals reach their respective locations safely and without tampering. Here, we will focus on securing how vehicles connect with the gateway (i.e., roadside infrastructure) by defining the security requirements for this particular use case, and its importance has already been discussed in earlier sections. The components and setups involved in the V2V use case scenario are explained in Section IV-B.

B. IDENTIFICATION OF CRITICAL/NON-CRITICAL ASSETS IN A V2V SCENARIO

In this section, the coverage of various critical and non-critical assets in V2V communication architecture and functional requirements will be discussed:

1) Assets: The assets can be physical components and non-physical components (like messages).
   a) MQTT-SN Publisher (ClientApp): An MQTT-SN publisher is an application that utilizes a ZigBee transceiver module to transmit distress signals to subscribers. This application might potentially be integrated into the vehicle's onboard automotive system. This application utilizes a ZigBee module transmitter that is operating in broadcast-sending mode. Every application possesses a distinct identification known as the ClientID, which may be derived from the vehicle. In this case, we can utilize the VIN number of the vehicles as the identifier. The ClientApp is developed using the React Native framework, which operates on the Samsung Galaxy Note 2 device equipped with 2GB of RAM and 32GB of storage. Additionally, it is connected to a plug-and-play ZigBee module for broadcasting messages. In our nomenclature, the ClientApp is represented by the subscript 'P'.
   b) MQTT-SN Forwarder (ForwardApp): An MQTT-SN forwarder is a program that receives MQTT-SN messages from clients and transmits them to the gateway using a ZigBee transceiver module. This application can easily be integrated into the onboard Android automotive system. Under typical conditions, the ZigBee module employed by this application operates in a state of readiness to receive signals. Upon receiving any message, it promptly disseminates it to the closest gateway or another MQTT-SN forwarder. The ForwardApp is developed using the React Native framework, which operates on a Samsung Galaxy S21 device

equipped with 2GB of RAM and 64GB of storage. It is connected to a plug-and-play ZigBee module for broadcasting messages. The ForwardApp is denoted by subscript ’F’ in our notation.

c) MQTT-SN Gateway (Roadside Infrastructure): An MQTT-SN gateway is a type of roadside infrastructure that accepts a message from a MQTT-SN client or forwarder over ZigBee and then forwards it to the MQTT broker. The gateway translates the MQTT-SN message into MQTT. The message is transmitted via the MQTT protocol using the TCP channel. The ZigBee module is in a state of actively monitoring roadside infrastructures to receive data from the MQTT-SN forwarder. The Roadside infrastructure is configured with the destination address of the MQTT broker, which serves as the control centre. The implementation utilizes a Raspberry Pi 3B+ module, namely the Broadcom BCM2837B0 quad-core A53 (ARMv8) processor. This module is equipped with a 32-bit board and includes both ZigBee and Wi-Fi capabilities. The roadside infrastructure is denoted by subscript ’G’ in our notation. There are already roadside infrastructures available in multiple countries. These can easily be programmed to support this architecture to make it less expensive.

d) MQTT Broker (Control Centre): The MQTT distress message received by the broker over the TCP channel from the MQTT-SN gateway. The control centre’s task is to distribute messages based on the subscriptions. Control centre runs on a laptop with specifications: M2 Pro Processor, 16GB RAM and 512GB hard disk. The Control centre is denoted by subscript ’B’ in our notation.

e) MQTT Subscriber (Emergency Services): The MQTT subscribers subscribe to various distress signal themes. The emergency services encompass law enforcement agencies, medical facilities, and vehicle recovery services. The emergency services operate on a laptop equipped with the following specifications: an 8th generation i7 processor, 16 gigabytes of RAM, and a 512 GB hard disc. The emergency services are denoted by subscript ’S’ in our notation.

f) Information/Data:
   - Vehicle Status Messages: It represents the current location of vehicle and other information necessary for emergency services.
   - ClientID: Unique vehicle information used to maintain sessions.
   - Topics: It represents the current state of the vehicle (i.e., vehicle not working, accident with or without injury) sent from the vehicle to respective emergency services.

2) Functional Requirements: There is one of the main functional requirements for V2V communication:
   a) Vehicle should send a status message to the end-user through the broker when there is an emergency.

C. THREAT MODEL FOR A V2V SCENARIO

In this section, threats to V2V communication use case are defined.

Threat 1

- Threat: The broker authorizes the adversary as if the adversary had a valid ClientID to send malicious status messages. [Authorization]
- Attack Name: Spoofing Attack
- Level of Attacker Difficulty: Medium
- Threat Severity: High
- Threat Probability: Medium
- Summary: The external attacker type guesses the unique ClientID (i.e. VIN) of a vehicle and publishes malicious status messages for the Subscriber (i.e., emergency services) through the broker. The objective of the attacker is to waste the resources of emergency services.
- Preconditions:
  – The attacker has access to a vehicle, which the attacker uses to deduce ClientID by guessing or reverse engineering based on the format used by manufacturers [50].
  – The attacker has an authorized account with the broker.
  – The attacker has an explicit knowledge of the structure and meaning of the ClientID and topics.
- Steps:
  1) Emergency Services: The emergency services subscribe to the topics published by the ClientApp.
  2) Adversary: The external attacker publishes all status messages under the topics linked Client ID of the ClientApp.
  3) Control Centre: The Control Centre forwards the information related to the topics linked to the ClientID of the vehicle to the emergency services subscribing to the topics.
- Postcondition: The attacker has successfully able to waste the resources of emergency services.

Threat 2

- Threat: Unauthorized individuals or entities may attempt to intercept and eavesdrop on V2V communications to obtain sensitive information, such as location, speed, or other private data. [Confidentiality]
- Attack Name: Passive Eavesdropping
- Level of Attacker Difficulty: Easy
- Threat Severity: Low
- Threat Probability: High
- Summary: The external attacker subscribes to the wildcard subscription (i.e., #) and gains access to all the messages published by the vehicle for the emergency

services through the broker. The objective of the attacker is to know the sensitive information.

- Preconditions:
  - The attacker has physical access to a vehicle, which is running ForwardApp.
  - The attacker has an authorized account with the Control Centre.
  - The attacker has explicit knowledge of the structure and meaning of the ClientID.
- Steps:
  1) Adversary: The external attacker subscribes to all the information linked to Client ID using a wildcard subscription. The attacker may or may not use guessed ClientID. The attacker may also listen to the ForwardApp messages.
  2) ClientApp: The ClientApp publishes the status message.
  3) Control Centre: The broker sends/publishes all the information it has related to all the topics to the adversary.
- Postcondition: The attacker can use the information to understand the vehicle details and end-user current information.

Threat 3

- Threat: Attackers can record and replay legitimate V2V messages, which can have negative consequences, especially when safety-critical information is involved. [Integrity]
- Attack Name: Replay
- Level of Attacker Difficulty: Low
- Threat Severity: Medium
- Threat Probability: Medium
- Summary: The external attacker can copy the packet published by the ClientApp and replay it back to the emergency services through a Control Centre, which will give false information to the emergency services. Thus, the attacker's objective is to misguide and waste the resources of emergency services.
- Preconditions:
  - The attacker has an authorized account with the Control Centre.
  - The attacker has the means to intercept and capture a message from the ClientApp to the Control Centre.
- Steps:
  1) ClientApp: The ClientApp publishes a control/status message to the Control Centre.
  2) Adversary: The external attacker intercepts the message and saves it. The external attacker publishes the saved message.
  3) Control Centre: The Control Centre sends the old message to all the emergency services subscribing to that topic.
- Postconditions:
  - The Control Centre does not recognize the old message.
  - The attacker can replay the status messages to exhaust the resources of emergency services.

Threat 4

- Threat: The attacker can intercept and modify messages to misguide the emergency services. [Integrity]
- Attack Name: Injection Attack
- Level of Attacker Difficulty: High
- Threat Severity: High
- Threat Probability: Low
- Summary: The external attacker can copy the message published by the ClientApp, modify it, and forward the modified message to the Control Centre for the emergency services, which will give false information to the emergency services in case of a status message. Thus, the attacker's objective is to misguide the emergency services.
- Preconditions:
  - The attacker has physical access to a ForwardApp.
  - The attacker has an authorized account with the Control Centre.
  - The attacker can intercept, capture, and modify a message from the ClientApp to the Control Centre.
- Steps:
  1) ClientApp: The ClientApp publishes a status message to the Control Centre.
  2) Adversary: The external attacker intercepts the message and modifies it. The external attacker publishes the modify message.
  3) Broker: The Control Centre sends/publishes the corrupted message to all the emergency services subscribing to that topic.
- Postconditions:
  - The Control Centre does not recognize the modified message.
  - The attacker can send the status message to misguide the emergency services.
  - The attacker can send the wrong status of ClientApp.

Threat 5

- Threat: Attackers can flood the V2V communication network with a high volume of messages, causing congestion and potentially disrupting the communication system, leading to communication failures [Availability]
- Attack Name: Denial of Service
- Level of Attacker Difficulty: Low
- Threat Severity: High
- Threat Probability: High
- Summary: An attacker can cause a denial-of-service attack by publishing a high volume of status messages. As a result, the resources of the Control Centre will exhaust.
- Preconditions:
  - The attacker has physical access to a similar ClientApp, which the attacker uses to send volumes of status messages
  - The attacker may have valid authentication with the Control Centre.
- Steps:

1) ClientApp: The ClientApp is connected with the broker under a Client ID.
2) Adversary: The attacker used the ClientID to connect with the same Control Centre.
3) Control Centre: The Control Centre accepts the connection from the adversary and sends high volumes of status messages to the emergency services.

- Postcondition: The attacker tries to make sure the emergency services are inaccessible to the end-user.

Threat 6

- Threat: Attackers may use jamming devices to disrupt the radio frequencies used by V2V communication, preventing legitimate vehicles from communicating with each other. [Availability]
- Attack Name: Jamming
- Level of Attacker Difficulty: High
- Threat Severity: High
- Threat Probability: Medium
- Summary: An attacker may cause a signal-jamming attack and stop ClientApp from sending any message to the emergency services.
- Preconditions:
  – The attacker should be present in the communication range of the ClientApp and ForwardApp.
  – The attacker may have access to the signal jammer device.
- Step: The attacker uses a signal jammer to stop the communication between ClientApp and road-side infrastructure, or Gateway and Control Centre.
- Postcondition: The attacker tries to make sure the emergency services are inaccessible to the end-user.

Threat 7

- Threat: Attackers may try to subscribe to the messages sent to control centre. [Authorization]
- Attack Name: Physical Attack
- Level of Attacker Difficulty: Low
- Threat Severity: High
- Threat Probability: High
- Summary: An attacker may subscribe to the messages sent by ClientApp to control centre for the emergency services.
- Precondition: The attacker should know the topics of the ClientApp publishing information.
- Step: The attacker sends a subscribe message to the control center with the known topics to receive the messages sent by the ClientApp.
- Postcondition: The attacker tries to provide the physical harm after getting all the information from control centre.

Threat 8

- Threat: Attackers may tamper the ClientApp. [Authorization]
- Attack Name: Denial of Service
- Level of Attacker Difficulty: High
- Threat Severity: High

TABLE 1. Quantification of Different Threat Model Parameters

ine **Parameter**	**Classification**	**Value**
Attacker's Difficulty	Low (easier for attacker)	3
	Medium	2
	High (harder for attacker)	1
Threat Severity	Low (not too severe)	1
	Medium	2
	High (Severe)	3
Threat Probability	Low (less likely)	1
	Medium	2
	High (more likely)	3

TABLE 2. Risk Assessment of All the Threats for V2V Communication

Threat No.	**Threat Severity (C)**	**Threat Probability (T)**	**Attacker's Difficulty (D)**	**Risk (R) = C x D x T**
Threat 1	3	2	2	12
Threat 2	1	3	3	9
Threat 3	2	2	3	12
Threat 4	3	1	1	3
Threat 5	3	3	3	27
Threat 6	3	2	1	6
Threat 7	3	3	3	27
Threat 8	3	1	1	3

- Threat Probability: Low
- Summary: An attacker may tamper with the ClientApp such that in case of emergencies it should not be able to send emergency messages to control centre.
- Precondition: The attacker should have physical access to the ClientApp.
- Step: The attacker may tamper with the ClientApp.
- Postcondition: The attacker tampers in such a way that ClientApp is unable to send any message.

D. SECURITY REQUIREMENTS FOR V2V USECASE SCENARIO

We have identified the above security objectives and threats based on the need for the V2V communication use case. Here, we try to define security requirements based on security objectives and the threat model defined earlier:

- SR1 - Data Origin Authentication: The broker transmits the information to the subscriber. A smartphone application, also known as a subscriber, should have the capability to offer the end-user a method, such as a digital signature or Message Authentication Code (MAC), to authenticate the origin of information. In the context of V2V communication, emergency services need to be able to authenticate the source of a status message received from a ClientApp.
- SR2 - Identification: The broker must verify the identity of an publisher and subscriber. In case of V2V communication, emergency services subscribing to the topics at

TABLE 3. Prioritizing Security Requirements of V2V Communication Based on Risk Assessment

Security Goal	Security Requirements	[Threat, Risk]	Total Risk = Sum (Ri)	Priority
Identification	SR2: Identification	[Threat 1, 12]	12	6
Authentication	SR1: Data Origin Authentication	[Threat 1, 12], [Threat 3, 12]	24	2
Confidentiality	SR3: Information Confidentiality	[Threat 2, 9] , [Threat 1, 12]	21	3
Integrity	SR5: Information Integrity (Payload)	[Threat 4, 3], [Threat 3, 12],	15	4
	SR6: Message Header Integrity	[Threat 3, 12], [Threat 4, 3]	15	5
Availability	SR7: Limited Publication	[Threat 6, 6]	6	7
Authorization	SR4: Access Control	[Threat 7, 27]	27	1
Physical Security	SR8: Physical Protection	[Threat 8, 3]	3	8

control centre and also the check identity of the publishers (i.e., ClientApp). We can use an unique identity which is only known to each vehicle (i.e., ClientApp)

- SR3 - Information Confidentiality: The publisher will employ cryptography, namely cryptographic techniques and key sizes, to ensure the confidentiality of the payload data throughout the whole communication process. The payload data must undergo encryption before reaching the destination, specifically the subscribers. Subsequently, the data remains encrypted at the Control Centre until it is transmitted to the emergency services.
- SR4 - Access Control: The broker should employ an access control mechanism so that any unauthorized user should not have access to message which they are not entitled for. In the case of V2V communication, this requirement is applicable to control centre. The control center should make sure proper emergency service should subscribe to right topic.
- SR5 - Information Integrity: The publisher transmits the information to the subscriber. A smartphone application should have the capability to offer the end-user the necessary tools (such as digital signature, MAC, and hashing) to identify any abnormalities in the MQTT-SN payload, including modification, deletion, insertion, replay, and other integrity issues, from the beginning to the end. In the context of V2V communication, emergency services should also have the measure to check the integrity of the status message received from the ClientApp.
- SR6 - Message Header Integrity: The publisher transmits the information to the gateway. A smartphone application should have the capability to offer the end-user the necessary tools (such as digital signature, MAC, and hashing) to identify any abnormalities in the MQTT-SN header, including modification, deletion, insertion, replay, and other integrity issues, that may occur during transmission between two points. Likewise, the message header transmitted by the publisher (ClientApp) must remain unchanged until it reaches the subscribers (Emergency Services). Publishers/subscribers can rely on the broker. Regarding V2V communication, emergency services can check the integrity of the status message header received from the ClientApp.
- SR7 - Limited Publication: The broker incorporates a security measure called "Handling Duplicate ClientID" to guarantee the continuous availability of services. Additionally, the gateway device includes functionality that checks the size of response messages to further ensure service availability. With respect to V2V communication, emergency services can eliminate duplicate status signals.
- SR8 - Physical Security: The end-devices should be protected in such a way that it should not be tampered with. In case of V2V communication, ClientApp should be protected against physical tampering.

V. ANALYSIS OF SECURITY REQUIREMENTS FOR A V2V SCENARIO

Our security requirements based on security goals have been categorized in this section. They have also been assigned priorities based on the threats addressed by different security requirements and their severity, probability and attacker's difficulty. Various threat severity, threat likelihood, and attacker difficulty metrics have been quantified to determine the risk based on attacker difficulty and give priority. In practical risk assessments, numerical figures for threat probability, severity, and attacker's difficulty are often not trusted enough. Therefore, a qualitative approach is commonly used to rate risks. As a result, a categorical rating system (low, medium, high) is utilized to determine the level of risk, as per the citation [48]. The formula below is taken from Chapter 1 of [48]. Rather focusing on the vulnerability, the formula has been modified to include the attacker's challenge in estimating the associated risk as follows:

$$R = C \times T \times D \quad (1)$$

where R, C, T and D are risk, threat severity, threat probability, and attacker's difficulty, respectively.

The three essential characteristics that are used to calculate risk have numerical values are shown in Table 1. The higher numerical value of a parameter value signifies greater risk

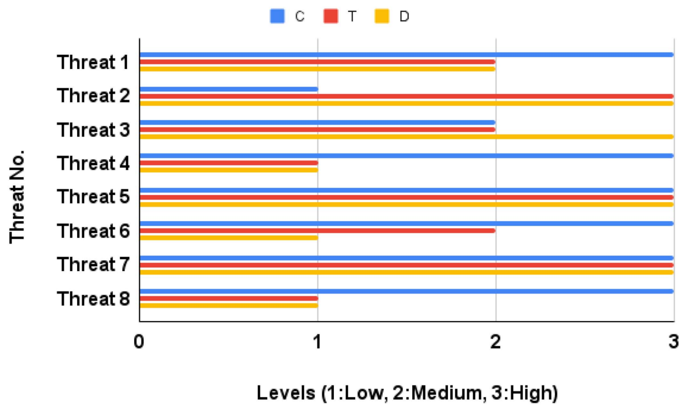

FIGURE 2. Threat vs C, T, and D.

than a lower numerical value. For example, in the case of an attacker's difficulty, if the attack is more straightforward for an adversary, the value is assigned three (i.e., more risk). On the other hand, if the attack is difficult for the attacker to execute, the value is assigned one (i.e., less risk). Similarly, for threat severity, if the threat is severe, it can cause more damage, and then the value is assigned three (i.e., more risk). Despite this, if the threat is less severe, which means minor damage to the system, then the value assigned is one (i.e., less risk). Furthermore, for threat probability, if the threat is more likely to occur in the attack, then the value is assigned three (i.e., more risk). However, if the threat is less likely to occur in the attack, the value assigned is one (i.e., less risk). In Table 2 and Fig. 2, all these three parameters will help us calculate the risk value for each threat mentioned in Section IV-C for V2V communication.

Table 3 illustrates which risks are being addressed by the security standards. We determine the overall risk that a security requirement addresses and then prioritize the requirements.

VI. ANALYSIS AND CONCLUSION

Through the utilization of the threat model and security requirements, it has been determined that the end-to-end security concept in the publish-subscribe protocol is more complex compared to the client-server protocol. The MQTT-SN packet was partitioned into two distinct elements: the payload containing the information and the message header. Given that each intermediate node needs only the MQTT-SN header to carry out its functions, a hybrid security model has been proposed for a publish-subscribe protocol that ensures both confidentiality and integrity. End-to-end security implementation for the MQTT-SN payload has been proposed. It is hard to provide end-to-end security where the publisher and subscriber do not know each other. Regarding the MQTT-SN header, point-to-point security should be put in place.

Additionally, it is necessary to provide an access control mechanism at either the publisher or subscriber level, as we cannot place faith in the broker, which may be a third-party program. In addition, we discovered that ensuring availability is a crucial security objective in the V2V communication use case, a concern that is often overlooked in numerous other IoT applications. Furthermore, the significance of employing a systematic approach to identify security requirements has been demonstrated. Hence, it underscores the significance of the security requirement process.

By utilizing the priorities outlined in Tables 2 and 3, one can develop a security framework that effectively addresses the requirements of each use case according to their respective priorities.

REFERENCES

[1] O. Alrawi, C. Lever, M. Antonakakis, and F. Monrose, “SoK: Security evaluation of home-based IoT deployments,” in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 1362–1380, doi: 10.1109/SP.2019.00013.

[2] P. Papadimitratos, “Secure vehicular communication systems,” in *Encyclopedia of Cryptography, Security and Privacy*. Berlin, Germany: Springer, 2024, pp. 1–6.

[3] H. Mun, M. Seo, and D. H. Lee, “Secure privacy-preserving V2V communication in 5G-V2X supporting network slicing,” *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 14439–14455, Sep. 2022.

[4] Y. Qian, K. Lu, and N. Moayeri, “A secure VANET MAC protocol for DSRC applications,” in *Proc. IEEE GLOBECOM 2008 IEEE Glob. Telecommun. Conf.*, 2008, pp. 1–5.

[5] S. Aarthi and N. Bharathi, “Analysis of security and privacy issues over vehicular communication in Internet of Vehicles,” in *Proc. IEEE 2022 Int. Mobile Embedded Technol. Conf.*, 2022, pp. 500–505.

[6] T. Yoshizawa et al., “A survey of security and privacy issues in V2X communication systems,” *ACM Comput. Surv.*, vol. 55, no. 9, pp. 1–36, 2023.

[7] J. Huang, D. Fang, Y. Qian, and R. Q. Hu, “Recent advances and challenges in security and privacy for V2X communications,” *IEEE Open J. Veh. Technol.*, vol. 1, pp. 244–266, 2020.

[8] B. Fabian, S. Gürses, M. Heisel, T. Santen, and H. Schmidt, “A comparison of security requirements engineering methods,” *Requirements Eng.*, vol. 15, pp. 7–40, 2010.

[9] D. Mellado, E. Fernández-Medina, and M. Piattini, “A common criteria based security requirements engineering process for the development of secure information systems,” *Comput. Standards Interfaces*, vol. 29, no. 2, pp. 244–253, 2007.

[10] N. R. Mead, “How to compare the Security Quality Requirements Engineering (SQUARE) method with other methods,” Softw. Eng. Inst., Carnegie-Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU/SEI-2007-TN-021, 2007.

[11] D. Muñante, V. Chiprianov, L. Gallon, and P. Aniorté, “A review of security requirements engineering methods with respect to risk analysis and model-driven engineering,” in *Proc. Availability, Rel., Secur. Inf. Syst.: IFIP WG 8.4, 8.9, TC 5 Int. Cross-Domain Conf. 2014 4th Int. Workshop Secur. Cogn. Informat. Homeland Defense 2014*, Fribourg, Switzerland, Sep. 2014, pp. 79–93.

[12] H. Gupta and A. Nayak, “Addressing IoT security challenges: A framework for determining security requirements of smart locks leveraging MQTT-SN,” in *Proc. IEEE Int. Symp. Netw., Comput. Commun.*, 2023, pp. 1–7, doi: 10.1109/ISNCC58260.2023.10323864.

[13] H. Gupta and A. Nayak, “Use of MQTT-SN in sending distress signals in vehicular communication,” in *Proc. IEEE Int. Symp. Netw., Comput. Commun.*, 2023, pp. 1–6, doi: 10.1109/ISNCC58260.2023.10323828.

[14] A. Stanford-Clark and H. L. Truong, “MQTT for sensor networks (MQTT-SN) protocol specification version 1.2,” IBM Corp., Endicott, NY, USA, Nov. 2013.

[15] M. Menzel, I. Thomas, and C. Meinel, “Security requirements specification in service-oriented business process management,” in *Proc. IEEE Int. Conf. Availability, Rel. Secur.*, 2009, pp. 41–48, doi: 10.1109/ARES.2009.90.

[16] Q. Zhao, L. Shu, K. Li, M. A. Ferrag, X. Liu, and Y. Li, “Security and privacy in solar insecticidal lamps Internet of Things: Requirements and challenges,” *IEEE/CAA J. Automatica Sinica*, vol. 11, no. 1, pp. 58–73, Jan. 2024, doi: 10.1109/JAS.2023.123870.

[17] S. Myagmar, A. J. Lee, and W. Yurcik, “Threat modeling as a basis for security requirements,” Univ. of Pittsburgh, Pittsburgh, PA, USA, Tech. Rep., 2005.

[18] D. Firesmith, "Engineering security requirements," *J. Object Technol.*, vol. 2, pp. 53–68, 2003, doi: 10.5381/jot.2003.2.1.c6.

[19] D. Firesmith, "Specifying reusable security requirements," *J. Object Technol.*, vol. 3, pp. 61–75, 2004, doi: 10.5381/jot.2004.3.1.c6.

[20] X. Huang, P. Craig, H. Lin, and Z. Yan, "SecIoT: A security framework for the Internet of Things," *Secur. Commun. Netw.*, vol. 9, pp. 3083–3094, 2016, doi: 10.1002/sec.1259.

[21] A. F. A. Rahman, M. Daud, and M. Z. Mohamad, "Securing sensor to cloud ecosystem using Internet of Things (IoT) security framework," in *Proc. Int. Conf. Internet Things Cloud Comput.*, 2016, pp. 1–5, doi: 10.1145/2896387.2906198.

[22] I. Alqassem, "Privacy and security requirements framework for the Internet of Things (IoT)," in *Proc. 36th Int. Conf. Softw. Eng.*, 2014, pp. 739–741, doi: 10.1145/2591062.2591201.

[23] M. Abomhara and G. M. Køien, "Security and privacy in the Internet of Things: Current status and open issues," in *Proc. IEEE Int. Conf. Privacy Secur. Mobile Syst.*, 2014, pp. 1–8, doi: 10.1109/PRISMS.2014.6970594.

[24] I. Alqassem and D. Svetinovic, "A taxonomy of security and privacy requirements for the Internet of Things (IoT)," in *Proc. IEEE Int. Conf. Ind. Eng. Eng. Manage.*, 2014, pp. 1244–1248, doi: 10.1109/IEEM.2014.7058837.

[25] H.-J. Kim, H.-S. Chang, J.-J. Suh, and T.-s. Shon, "A study on device security in IoT convergence," in *Proc. IEEE Int. Conf. Ind. Eng., Manage. Sci. Appl.*, 2016, pp. 1–4, doi: 10.1109/ICIMSA.2016.7503989.

[26] S.-R. Oh and Y.-G. Kim, "Security requirements analysis for the IoT," in *Proc. IEEE Int. Conf. Platform Technol. Serv.*, 2017, pp. 1–6, doi: 10.1109/PlatCon.2017.7883727.

[27] K. Pelz, "Robustness of publish/subscribe systems: A survey," Dept. Telecommunication System, Technical Univ. Berlin, Berlin, Germany, Tech. Rep., 2020.

[28] Y. Liu and B. Plale, "Survey of publish subscribe event systems," Computer Science Dept., Indiana Univ. Bloomington, Bloomington, IN, USA, Tech. Rep., 2003.

[29] R. Strom et al., "Gryphon: An information flow based approach to message brokering," 1998, *arXiv:cs/9810019*.

[30] C. Wang, A. Carzaniga, D. Evans, and A. Wolf, "Security issues and requirements for internet-scale publish-subscribe systems," in *Proc. IEEE 35th Annu. Hawaii Int. Conf. Syst. Sci.*, 2002, pp. 3940–3947, doi: 10.1109/HICSS.2002.994531.

[31] D. Lagutin, K. Visala, A. Zahemszky, T. Burbridge, and G. F. Marias, "Roles and security in a publish/subscribe network architecture," in *Proc. IEEE Symp. Comput. Commun.*, 2010, pp. 68–74, doi: 10.1109/ISCC.2010.5546746.

[32] M. Ammar, G. Russello, and B. Crispo, "Internet of Things: A survey on the security of IoT frameworks," *J. Inf. Secur. Appl.*, vol. 38, pp. 8–27, 2018, doi: 10.1016/j.jisa.2017.11.002.

[33] C. Bormann, M. Ersue, and A. Keränen, "Terminology for constrained-node networks," RFC 7228, May 2014. [Online]. Available: https://rfc-editor.org/rfc/rfc7228.txt

[34] T. Kao, H. Wang, and J. Li, "Safe MQTT-SN: A lightweight secure encrypted communication in IoT," *J. Phys.: Conf. Ser.*, vol. 2020, 2021, Art. no. 012044, doi: 10.1088/1742-6596/2020/1/012044.

[35] J. Roldán-Gómez, J. Carrillo-Mondéjar, J. M. C. Gómez, and S. Ruiz-Villafranca, "Security analysis of the MQTT-SN protocol for the Internet of Things," *Appl. Sci.*, vol. 12, no. 21, 2022, Art. no. 10991, doi: 10.3390/app122110991.

[36] J. Roldán-Gómez, J. Carrillo-Mondéjar, J. M. C. Gómez, and J. L. M. Martínez, "Security assessment of the MQTT-SN protocol for the Internet of Things," *J. Phys.: Conf. Ser.*, vol. 2224, no. 1, Apr. 2022, Art. no. 012079, doi: 10.1088/1742-6596/2224/1/012079.

[37] A. Munshi, "Improved MQTT secure transmission flags in smart homes," *Sensors*, vol. 22, no. 6, 2022, Art. no. 2174, doi: 10.3390/s22062174.

[38] S. Farahani, *ZigBee Wireless Networks and Transceivers*. Boston, MA, USA: Newnes Publisher, 2008.

[39] M. Husnain et al., "Preventing MQTT vulnerabilities using IoT-enabled intrusion detection system," *Sensors*, vol. 22, no. 2, 2022, Art. no. 567, doi: 10.3390/s22020567.

[40] H. Sochor, F. Ferrarotti, and R. Ramler, "Exploiting MQTT-SN for distributed reflection denial-of-service attacks," in *Proc. Int. Conf. Database Expert Syst. Appl.*, 2020, pp. 74–81, doi: 10.1007/978-3-030-59028-47.

[41] O. Sadio, I. Ngom, and C. Lishou, "Lightweight security scheme for MQTT/MQTT-SN protocol," in *Proc. IEEE 6th Int. Conf. Internet Things: Syst., Manage. Secur.*, 2019, pp. 119–123, doi: 10.1109/IOTSMS48152.2019.8939177.

[42] F. Chen, Y. Huo, J. Zhu, and D. Fan, "A review on the study on MQTT security challenge," in *Proc. IEEE Int. Conf. Smart Cloud*, 2020, pp. 128–133, doi: 10.1109/SmartCloud49737.2020.00032.

[43] S. Andy, B. Rahardjo, and B. Hanindhito, "Attack scenarios and security analysis of MQTT communication protocol in IoT system," in *Proc. IEEE 4th Int. Conf. Elect. Eng., Comput. Sci. Inform.*, 2017, pp. 1–6, doi: 10.1109/EECSI.2017.8239179.

[44] S. Jabeen and S. R. Potturu, "Survey on security and privacy of connected vehicles and cloud platforms for communication," *AIP Conf. Proc.*, vol. 2512, 2024, Art. no. 020041.

[45] Y. Cao and N. Wang, "Toward efficient electric-vehicle charging using VANET-Based information dissemination," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 2886–2901, Apr. 2017, doi: 10.1109/TVT.2016.2594241.

[46] Y. Cao, N. Wang, and G. Kamel, "A publish/subscribe communication framework for managing electric vehicle charging," in *Proc. IEEE Int. Conf. Connected Veh. Expo*, 2014, pp. 318–324, doi: 10.1109/ICCVE.2014.7297564.

[47] S. Taghizadeh, P. Jamborsalamati, M. J. Hossain, and J. Lu, "Design and implementation of an advanced vehicle-to-vehicle (V2V) power transfer operation using communications," in *Proc. IEEE Int. Conf. Environ. Elect. Eng. 2018, IEEE Ind. Commercial Power Syst. Europe*, 2018, pp. 1–6, doi: 10.1109/EEEIC.2018.8494480.

[48] P. C. van Oorschot, *Computer Security and the Internet Security: Tools and Jewels*. Cham, Switzerland: Springer Nature, 2019.

[49] J. R. Vacca, *Computer and Information Security Handbook*. Boston, MA, USA: Newnes, 2012.

[50] Y. Jia et al., "Burglars' IoT paradise: Understanding and mitigating security risks of general messaging protocols on IoT clouds," in *Proc. IEEE Symp. Secur. Privacy*, 2020, pp. 465–481, doi: 10.1109/SP40000.2020.00051.

HEMANT GUPTA received the M.Tech. degree in software engineering from Birla Institute of Technology And Science - Pilani, Pilani, India, in 2015, and the M.Sc. degree in computer science from Carleton University, Ottawa, ON, Canada, in 2019. He is currently working toward the Ph.D. degree with the School of Electrical Engineering & Computer Science, Ottawa. He has more than seven years of industrial experience in the field of networking and vehicles. His research interests include network security, vehicular networks and IoT.

AMIYA NAYAK (Senior Member, IEEE) received the B.Math. degree in computer science and combinatorics and optimization from University of Waterloo, ON, Canada, in 1981, and the Ph.D. degree in systems and computer engineering from Carleton University, Ottawa, ON, in 1991. He is currently a Full Professor with the School of Electrical Engineering and Computer Science at the University of Ottawa. He has more than 17 years of industrial experience in software engineering, avionics and navigation systems, simulation and system level performance analysis. His research interests include software-defined networking, mobile computing, wireless sensor networks, and vehicular ad hoc networks. He is now in the Editorial Board of IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE INTERNET OF THINGS JOURNAL, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE OPEN JOURNAL OF THE COMPUTER SOCIETY, *Future Internet*, and *International Journal of Distributed Sensor Networks*. He was in the Editorial Board of several journals, including IEEE TRANSACTIONS ON PARALLEL & DISTRIBUTED SYSTEMS, *International Journal of Parallel, Emergent and Distributed Systems*, *Journal of Sensor and Actuator Networks*, and *EURASIP Journal of Wireless Communications and Networking*.